ipd4300matxtrmTES-10

Defense Information Infrastructure (DII)
Common Operating Environment (COE)

Application Program Interface Reference Manual (APIRM)
for the
Textual Observation API (MATXT) Segment
of the
Tactical Environmental Support System Next Century
[TESS(NC)]
Meteorology and Oceanography (METOC) Database

Document Version 4.3

15 October 1998

Prepared for:
Naval Research Laboratory
Marine Meteorology Division
Monterey, CA

Prepared by:
Integrated Performance Decisions
Middletown, RI

Table of Contents

1 SCOPE 1
1.1 Identification 1
1.2 System Overview 1

2 REFERENCED DOCUMENTS 5
2.1 Government Documents 5
2.2 Non-Government Documents 7

3 MATXT OVERVIEW 9
3.1 Overview of the Textual Observation Database (MDTXT) Segment 9
3.2 API Overview 11
3.3 Supersession of TEDS APIs 13
3.4 Textual Observation Data Structures 14
3.4.1 Textual Observation Area Of Interest (AOI) Structure 15
3.4.2 Textual Observation Linked List Structure 15
3.4.3 Textual Observation Structure 15
3.4.4 Textual Observation Catalog Query Structure 16
3.4.5 Textual Observation Catalog Data Structure 16
3.4.6 Textual Observation Return Structure 17
3.5 MATXT Error Definitions 17

4 CONNECT APIS 19
4.1 MATXTConnect 20
4.2 MATXTDisconnect 21
4.3 MATXTRemoteConnect 22
4.4 MATXTRemoteDisconnect 23
4.5 MATXTSetConnection 24

5 RETRIEVAL APIS 25
5.1 MATXTCatalog 25
5.2 MATXTGetByQuery 26
5.3 MATXTGetByID 27

6 DATA MANAGEMENT APIS 29
6.1 MATXTIngest 29
6.2 MATXTUpdateByID 30
6.3 MATXTDeleteByID 31
6.4 MATXTDeleteByQuery 32

7 MATXT UTILITY METHODS AND FUNCTIONS 33
7.1 MATXTFreeLL 33

8 NOTES 35
8.1 Glossary of Acronyms 35

List of Tables

3-1 Textual Observation Data Types....9
3-2 Textual Observation Data APIs....12
3-3 MATXT and TEDS API Cross-Reference....13

List of Figures

1-1 TESS(NC) METOC Database Conceptual Organization....3

1 SCOPE

1.1 Identification

This Application Program Interface (API) Reference Manual (APIRM) describes the APIs provided in the Textual Observation API (MATXT) Segment, Version 4.2, of the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database. The MATXT segment provides APIs for the storage, retrieval, and manipulation of textual METOC observations and bulletins. This software is designed to run under the Defense Information Infrastructure (DII) Common Operating Environment (COE), release 3.1, on a Hewlett-Packard computer running HP-UX 10.20 or a personal computer running the Microsoft Windows NT 4.0 operating system with Service Pack 3.

1.2 System Overview

The APIs described in this document form a portion of the METOC Database component of the TESS(NC) Program (Navy Integrated Tactical Environmental Subsystem (NITES) Version I). On 29 October 1996, the Oceanographer of the Navy issued a TESS Program Policy statement in letter 3140 Serial 961/6U570953, modifying the Program by calling for five seamless software versions that are DII COE compliant, preferably to level 5.

The five versions are:

- NITES Version I — The local data fusion center and principal METOC analysis and forecast system (TESS(NC))
- NITES Version II — The subsystem on the Joint Maritime Command Information System (JMCIS) or Global Command and Control System (GCCS) (NITES/Joint METOC Segment (JMS))
- NITES Version III — The unclassified aviation forecast, briefing, and display subsystem tailored to Naval METOC shore activities (currently satisfied by the Meteorological Integrated Data Display System (MIDDS))
- NITES Version IV — The Portable subsystem composed of independent PCs/workstations and modules for forecaster, satellite, communications, and Integrated Command, Control, Communications, Computer, and Intelligence Surveillance Reconnaissance (IC4ISR) functions (currently the Interim Mobile Oceanographic Support System (IMOSS))
- NITES Version V — Foreign Military Sales (currently satisfied by the Allied Environmental Support System (AESS))

NITES I acquires and assimilates various METOC data for use by US Navy and Marine Corps weather forecasters and tactical planners. NITES I provides these users with METOC data, products, and applications necessary to support the warfighter in tactical operations and decision making. NITES I provides METOC data and products to NITES I and II applications, as well as non-TESS(NC) systems requiring METOC data, in a heterogeneous, networked computing environment.

The TESS(NC) Concept of Operations and system architecture require that the METOC Database be distributed both in terms of application access to METOC data and products and in terms of physical location of the data repositories. The organizational structure of the database is influenced by these requirements, and the components of this distributed database are described below.

In accordance with DII COE database concepts, the METOC Database is composed of six DII COE-compliant *shared database* segments. Associated with each shared database segment is an API segment. The segments are arranged by data type as follows:

Data Type	**Data Segment**	**API Segment**
Grid Fields	MDGRID	MAGRID
Latitude-Longitude-Time (LLT) Observations	MDLLT	MALLT
Textual Observations and Bulletins	MDTXT	MATXT
Remotely Sensed Data	MDREM	MAREM
Imagery	MDIMG	MAIMG
Climatology Data	MDCLIM	MACLIM

A typical client-server installation is depicted in Figure 1-1 on the next page. This shows the shared database segments residing on a DII COE SHADE database server, with a NITES I or II client machine hosting the API segments. Communication between API segments and shared database segments is accomplished over the network using ANSI-standard Structured Query Language (SQL).

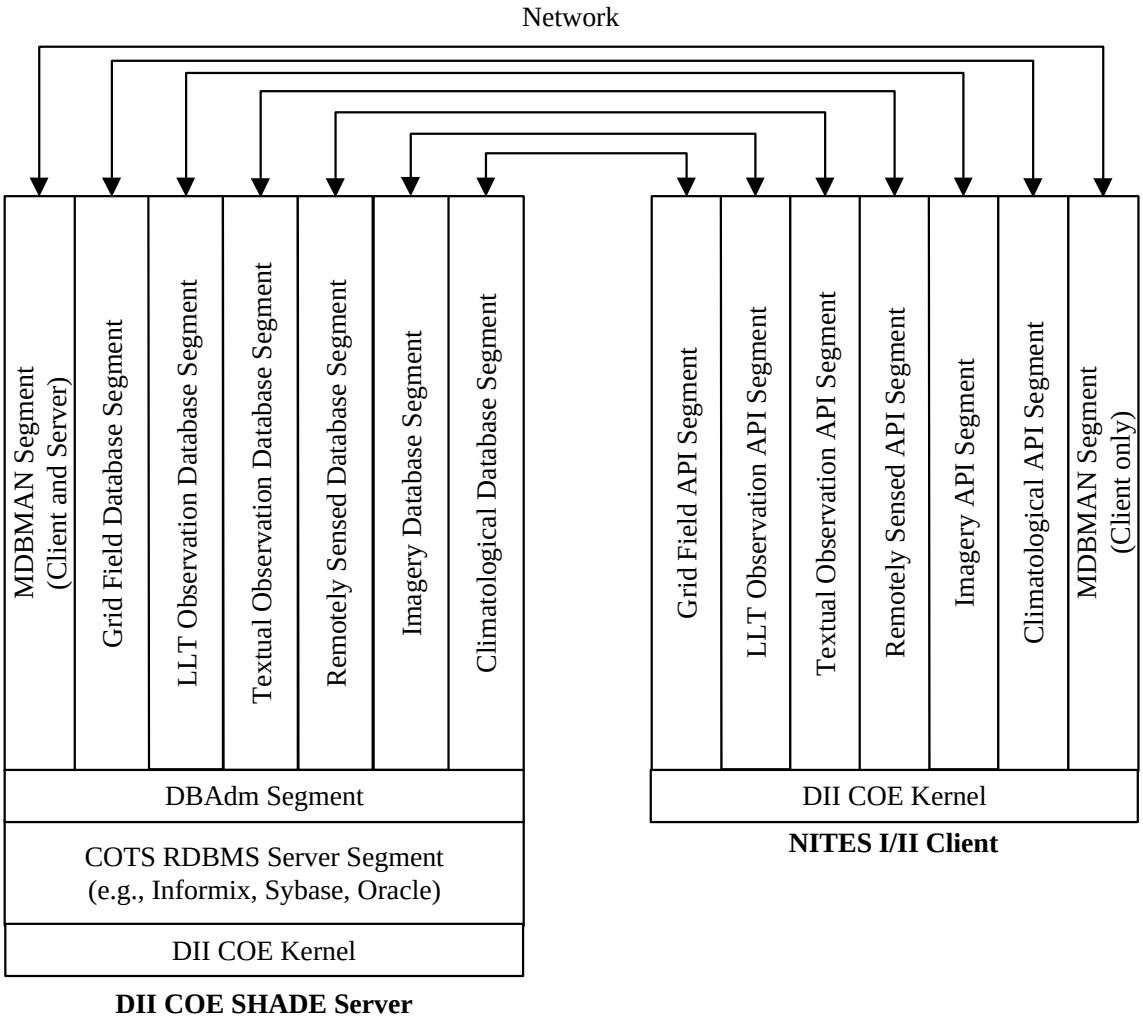

Figure 1-1. TESS(NC) METOC Database Conceptual Organization

The APIs in the MATXT segment deals with textual observations and bulletins. Textual observational data are primarily ASCII-formatted forecasts or bulletin/warning oriented messages. Textual observation data can be associated with a specific geographic point and time, but more generally are associated with a geographical area or region. Types include Forecast Reports, Warnings, and Notices. Depending on the type of textual observation, the reporting station or organization and the area or region affected is decoded and stored along with the textual portion of the message. Textual observation data are typically displayed as text by a client application.

(This page intentionally left blank.)

2 REFERENCED DOCUMENTS

2.1 Government Documents

STANDARDS

MIL-STD-498 5 December 1994	*Software Development and Documentation*

SPECIFICATIONS

DII COE I&RTS July 1997	*Defense Information Infrastructure (DII) Common Operating Environment (COE) Integration and Runtime Specification, Preliminary, Version 3.0*
Unnumbered 5 December 1997	*Performance Specification (PS) for the Tactical Environmental Support System/Next Century TESS(3)/NC (AN/UMK-3)*
Unnumbered 30 September 1997	*Software Requirements Specification for the Tactical Environmental Support System/Next Century [TESS(3)/NC] Meteorological and Oceanographic (METOC) Database*, Space and Naval Warfare Systems Command, Environmental Systems Program Office (SPAWAR PMW-185), Washington, DC

OTHER DOCUMENTS

Unnumbered 30 September 1997	*Database Design Description for the Tactical Environmental Support System/Next Century [TESS(3)/NC)] Meteorological and Oceanographic (METOC) Database*, Space and Naval Warfare Systems Command, Environmental Systems Program Office (SPAWAR PMW-185), Washington, DC
ipd4200matxtsvdTES-10 15 October 1998	*Software Version Description (SVD) for the Textual Observation API (MATXT) Segmentof the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database*

DII.COE.DocReqs-5
29 April 1997 — *Defense Information Infrastructure (DII) Common Operating Environment (COE) Developer Documentation Requirements, Version 1.0*

Office of the Assistant Secretary of Defense (Command, Control, Communications, and Intelligence), Washington, DC

DoD 8320.1-M-1
19 November 1996 — *Data Standardization Procedures (Draft)*

Commander, Naval Meteorology and Oceanography Command

COMNAVMETOCCOMINST 3141.2 — *Surface METAR Observations Users Manual*

COMNAVMETOCCOMINST 3144.1D — *Ship Weather Observations Manual*

Department of the Air Force, Headquarters Air Weather Service, Scott AFB, ILL

AWSR 105-2
24 August 1990 — *Weather Communications Policies and Procedures*

Naval Research Laboratory, Marine Meteorology Division, Monterey, CA

ipd4300matxtpmTES-10
15 October 1998 — *Programming Manual (PM) for the Textual Observation API (MATXT) Segment of the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database*

ipd4300matxtipTES-10
15 October 1998 — *Installation Procedures (IP) for the Textual Observation API (MATXT) Segment of the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database*

Space and Naval Warfare Systems Command, Environmental Systems Program Office (SPAWAR PMW-185), Washington, DC

Unnumbered
20 June 1997 — *Tactical Environmental Data System (TEDS) Release 3.5 Observation/Profile Data Applications Program Interface (API) User's Guide*

Office of the Federal Coordinator for Meteorological Services and Supporting Research, Washington, DC

FMH-1 December 1995	*Federal Meteorological Handbook No. 1: Surface Weather Observations and Reports*
FMH-2 December 1988	*Federal Meteorological Handbook No. 2: Surface Synoptic Codes*
FMH-10 December 1988	*Federal Meteorological Handbook No. 10: Meteorological Rocket Observations*
FMH-11 June 1991	*Federal Meteorological Handbook No. 11: Doppler Radar Meteorological Observations*

2.2 Non-Government Documents

World Meteorological Organization, Geneva, Switzerland

WMO-306 1995	*Manual On Codes*
WMO-386 1992	*Manual on the Global Telecommunications System*

(This page intentionally left blank.)

3 MATXT OVERVIEW

3.1 Overview of the Textual Observation Database (MDTXT) Segment

The Textual Observation database was designed using an object/relational hybrid method. The textual observation data are modeled as a class from a common root class. However, the methods are not associated with each class; they are associated with the Textual Observation database in general. The purpose of the Textual Observation APIs is to manage all aspects of textual meteorological observation of various types (forecasts, reports, warnings, etc.).

Table 3-1 below shows the types and subtypes of textual observations handled by MATXT. This information is derived from WMO-386, Volume I, Part II, Attachment II-5, Tables A and B1, and from AWSR 105-2 (for U.S. Air Force - specified bulletins).

Table 3-1. Textual Observation Data Types

Type	Type (Table A) Designator	Subtype (Table B1) Designator
Forecast Reports	F	E = Extended Forecast H = Upper Air Thickness I = Iceberg J = Radio Warning Service K = Tropical Cyclone Advisories L = Local Area Forecasts M = Temperature Extremes O = Guidance P = Public Q = Other Shipping V = Volcanic Ash W = Winter Sports X = Miscellaneous
Surface Reports	S	D = Radar Report (Parts A and B) T = Sea Ice U = Snow Depth X = Miscellaneous
Gridded	H	F = Aerological Diagrams
Pictorial	P	D = Thickness Data (Relative Topography)
Upper Air	U	X = Miscellaneous

Table 3-1. Textual Observation Data Types

Type	Type (Table A) Designator	Subtype (Table B1) Designator
Warnings	W	A = AIRMET/SIGMET C = Tropical Cyclone (SIGMET) D = Tropical Cyclone Discussion E = Tsunami F = Tornado (USAF) G = River Flood H = Hurricane M = High Seas (USAF) O = Other S = SIGMET T = Tropical Cyclone (Typhoon) U = Severe Thunderstorm V = Volcanic Ash (SIGMET) W = Military Weather Warnings (USAF) X = Misc. Weather Warnings (USAF)
Notices	N	G = Hydrological H = Marine N = Nuclear Emergency O = METNO/WIFMA P = Product generation delay T= Test Message W = Warning Related or Cancellation

3.2 API Overview

The methods are associated with the database in general and not each observation type. The purpose of this is to use the relational functionality provided by the RDBMS to implement the data model. There are seven methods (some of whose names have been changed since the preliminary (developer) release. In the list that follows, the new routine name is given first, then the old routine name. Applications using the developer's release routine names should be upgraded to the new routine names, as the old names will be phased out in the next release. For this release, both routine names will continue to function.

1. **MATXTIngest**: Adds an object to the database.
2. **MATXTDeleteByID, MATXTDeleteByQuery**: Deletes objects from the database based on a given criteria.
3. **MATXTCatalog**: Provides summary information about observations in the database.
4. **MATXTGetByID**: Retrieves a single object returned as a result of a catalog.
5. **MATXTGetByQuery**: Retrieves multiple objects.
6. **MATXTUpdateByID**: Updates limited attributes within an object.
7. **MATXTFreeLL**: Used to free a linked list returned by a query operation.

Each of the methods, with the exception of the Ingest methods, allows the input of selection criteria. The criteria are used to construct SQL statements to retrieve or otherwise manipulate objects which match the criteria. The Ingest methods do not provide any criteria because they are simply writing data to the database.

Five other methods are also provided to manage database connections:

1. **MATXTConnect**: Connects to the default Textual database.
2. **MATXTRemoteConnect**: Connects to the specified Textual database.
3. **MATXTDisconnect**: Disconnects from the default Textual database.
4. **MATXTRemoteDisconnect**: Disconnects from the specified Textual database.
5. **MATXTSetConnection**: Sets the current connection to the specified Textual database.

Table 3-2 provides a listing of all APIs for Textual observation data.

Table 3-2. Textual Observation Data APIs

API Name	Functional Description
MATXTCatalog	Catalogs observations based on a selection criteria.
MATXTConnect	Connects to the default Textual Observation Database.
MATXTDeleteByID, MATXTDeleteByQuery	Deletes an observation from the database.
MATXTDisconnect	Disconnects from Textual Observation Database.
MATXTGetByID	Gets a single observation based on information returned from a catalog query.
MATXTGetByQuery	Retrieves groups of observations based on selection criteria.
MATXTIngest	Adds an observation to the database.
MATXTFreeLL	Frees a linked list returned by a query operation.
MATXTRemoteConnect	Connects to the specified Textual Observation Database.
MATXTRemoteDisconnect	Disconnects to the specified Textual Observation Database.
MATXTConnect	Sets the active connection to the specified Textual Observation Database.
MATXTUpdateByID	Updates an observation based on given criteria.

3.3 Supersession of TEDS APIs

3.3.1 MATXT and TEDS 3.5 API Cross-Reference

The MATXT APIs will, when implemented, replace the older Tactical Environmental Data System (TEDS) APIs, documented in the *TEDS Release 3.5 Observation/Profile API User's Guide* referenced in Section 2 of this document. Applications using the TEDS APIs will need to be rewritten to use the equivalent MATXT APIs. Table 3-3 provides a cross-reference of the MATXT and TEDS APIs for the convenience of developers.

The following important considerations should be noted:

- In some cases, a single MATXT API replaces several TEDS APIs.
- Some of the APIs in the *TEDS Release 3.5 Observation/Profile API User's Guide* have been placed in other segments of the TESS(NC) METOC Database. The APIs for the decoded LLT observations and bulletins are in the MALLT segment, and those for vertical soundings and Special Sensor Microwave/Imager will be in the MAREM segment.
- Because the data structures differ between the two implementations, it is imperative that care be taken to ensure correct conversion.

Table 3-3. MATXT and TEDS API Cross-Reference

TEDS API(s)	MATXT API
ted_AddTXT	MATXTIngest
ted_GetTXTCatalog	MATXTCatalog
ted_start	MATXTConnect
ted_stop	MATXTDisconnect
ted_GetTXT	MATXTGetByID
ted_GetTXTs	MATXTGetByQuery
ted_DeleteTXT	MATXTDeleteByID
ted_UpdateTXT	MATXTUpdateByID

3.3.2 Obsolete APIs

The following list of APIs will not be supported in the next delivery of the MATXT segment. References to these APIs have been removed from the APIRM. The libraries will continue to support these APIs until the next delivery.

Obsolete		Superseded By
MATXTGetCatalog	–>	MATXTCatalog
MATXTGet	–>	MATXTGetByQuery
MATXTGetByRef	–>	MATXTGetByID
MATXTAdd	–>	MATXTIngest
MATXTUpdate	–>	MATXTUpdateByID
MATXTDelete	–>	MATXTDeleteByID

3.4 Textual Observation Data Structures

The following structures are used by the Textual Observation APIs and are provided for reference.

3.4.1 Textual Observation Area Of Interest (AOI) Structure

This structure contains the boundaries of an AOI.

```
typedef struct tagmatxtaoi {
      float       rsNorthLat;             /* Northernmost latitude          */
      float       rsSouthLat;             /* Southernmost latitude          */
      float       rsEastLon;              /* Easternmost longitude          */
      float       rsWestLon;              /* Westernmost longitude          */
} MATXTAOI, *PMATXTAOI;
```

3.4.2 Textual Observation Linked List Structure

This structure contains a linked list of textual observation catalog data or textual observation data structures.

```
typedef struct tagmatxtlinkedlist {
      void      *pNext;                   /* Pointer to next element        */
      void      *pPrev;                   /* Pointer to previous element    */
      void      *pData;                   /* Pointer to the data            */
                                          /* structure                      */
      int       nInternalMask;            /* Internal use only              */
} MATXTLINKEDLIST, *PMATXTLINKEDLIST;
```

3.4.3 Textual Observation Structure

This structure contains the data for a single textual observation. All times are epoch times (seconds since 0000Z 1 January 1970).

```
typedef struct tagmatxtobs {
   char       chType;                   /* Observation type defined       */
                                        /* in TESS/NC METOC DB SRS        */
                                        /* Section 3.2.3.11 and           */
                                        /* WMO-386 Table A.               */
   char       chSubType;                /* Observation subtype (see       */
                                        /* TESS/NC METOC DB SRS           */
                                        /* Section 3.2.3.11 and           */
                                        /* WMO-386 Table B1.              */
   int        nQualityIndicator;        /* Quality indicator.  0          */
                                        /* indicates data passed          */
                                        /* quality checks.                */
   long       lReportTime;              /* Report time                    */
   long       lBegValidTime;            /* Earliest valid time            */
                                        /* If unknown set to 0            */
   long       lEndValidTime;            /* Latest valid time              */
                                        /* If unknown set to 0            */
```

```
   long       lReceiptTime;              /* Time the message was          */
                                         /* received by the database      */
   MATXTAOI   matxtAOI;                  /* Area of interest              */
   int        nDataCategory;             /* Category of data              */
                                         /* 0 = Base, 1 = Derived,        */
                                         /* 2 = Edited                    */
   char       szSecurityClass[32];       /* Security classification       */
   char       szOriginatingSite[32];     /* Originating site name         */
   char       szReceiptMethod[32];       /* Receipt method                */
                                         /* (e.g., COMEDS)                */
   int        nDataSize;                 /* Size of data (bytes)          */
   char       *pData;                    /* Text of message               */
} MATXTOBS, *PMATXTOBS;
```

3.4.4 Textual Observation Catalog Query Structure

This structure is used to submit a catalog query. Those portions that are filled in constitute the query criteria. All times are epoch times (seconds since 0000Z 1 January 1970).

```
typedef struct tagmatxtcatquery {
   char       chType;                    /* Textual observation type      */
                                         /* defined in TESS/NC METOC      */
                                         /* DB SRS Section 3.2.3.11       */
                                         /* and WMO-306 Table A.          */
   char       chSubType;                 /* Subtype defined in TESS/      */
                                         /* NC METOC DB SRS Section       */
                                         /* 3.2.3.11 and WMO-306          */
                                         /* Table B1                      */
   long       lBegReportTime;            /* Beginning time of report.     */
                                         /* Used in query.  0 is wildcard.*/
                                         /* Used in query.  0 is wildcard.*/
   long       lEndReportTime;            /* Ending time of report.        */
                                         /* Used in query.  0 is wildcard.*/
   long       lBegValidTime;             /* Earliest valid time           */
                                         /* Used in query.  0 is wildcard.*/
   long       lEndValidTime;             /* Latest valid time             */
                                         /* Used in query.  0 is wildcard.*/
   long       lBegReceiptTime;           /* Beginning time range for the  */
                                         /* receipt time.  Used in the    */
                                         /* query.  0 is wildcard.        */
   long       lEndReceiptTime;           /* Ending time range for the     */
                                         /* receipt time.  Used in the    */
                                         /* query.  0 is wildcard.        */
   MATXTAOI   matxtAOI;                  /* Area of interest              */
   char       szOriginatingSite[32];     /* Originating site name         */
   char       szReceiptMethod[32];       /* Receipt method                */
} MATXTCATQUERY, *PMATXTCATQUERY;
```

3.4.5 Textual Observation Catalog Data Structure

This structure contains the catalog data for a single textual observation record. All times are epoch times (seconds since 0000Z 1 January 1970).

```
typedef struct tagmatxtcatdata {
   int         nObjectID;                /* Unique object identifier       */
   char        chType;                   /* Observation type defined       */
                                         /* in TESS/NC METOC DB SRS        */
                                         /* Section 3.2.3.11 and           */
                                         /* WMO-306 Table A.               */
   char        chType;                   /* Type of the textual ob.        */
                                         /* Defined in TESS/NC SRS         */
                                         /* Section 3.2.3.11 and in        */
                                         /* WMO-386 Table A.               */
   char        chSubType;                /* Subtype defined in TESS/       */
                                         /* NC METOC DB SRS Section        */
                                         /* 3.2.3.11 and WMO-306           */
                                         /* Table B1                       */
   char        szSubTypeString[32];      /* Meaningful character string    */
                                         /* for the subtype.               */
   long        lReportTime;              /* Report time                    */
   long        lBegValidTime;            /* Earliest valid time            */
   long        lEndValidTime;            /* Latest valid time              */
   long        1ReceiptTime;             /* Time the message was           */
                                         /* received by the database       */
   MATXTAOI    matxtAOI;                 /* Area covered by report         */
   char        szOriginatingSite[32];    /* Originating site name          */
   char        szReceiptMethod[32];      /* Receipt method                 */
} MATXTCATDATA, *PMATXTCATDATA;
```

3.4.6 Textual Observation Return Structure

Each of the textual observation APIs returns this structure containing status data. The *nStatus* field will contain a zero upon successful completion of the API call. If the field is set to 1, there is an SQL error, and the *szSQLState* field must be examined. Any value of *nStatus* greater than 1 indicates an MATXT error that maps to a define in the file **MATXTErr.h**.

```
typedef struct tagmatxtret {
   int    nStatus;           /* nStatus = 0 ==> Success                   */
                             /* nStatus = 1 ==> Check SQLState            */
                             /* nStatus >=2 ==> Segment-specific          */
                             /* error defined in MATXTErr.h               */
   char   szSQLState[6];     /* First 2 characters represent the          */
                             /* class, last 3 characters the              */
                             /* subclass where an SQL error               */
                             /* occurred                                  */
   char   szErrorMessage[290];
                             /* The actual error message, of form         */
                             /* “subroutine: message”                     */
} MATXTRET, *PMATXTRET;
```

3.5 MATXT Error Definitions

The following are the MATXT error definitions contained in the file **MATXTErr.h**.

```
#define     MATXTERR_OFFSET                                    8000

#define     MATXT_MIN_STATUS                      1 + MATXTERR_OFFSET
#define     MATXT_MAX_STATUS                     16 + MATXTERR_OFFSET

#define     MATXT_NULLPOINTER                     1 + MATXTERR_OFFSET
#define     MATXT_INVALIDTYPE                     2 + MATXTERR_OFFSET
#define     MATXT_INVALIDSUBTYPE                  3 + MATXTERR_OFFSET
#define     MATXT_INVALIDTIME                     4 + MATXTERR_OFFSET
#define     MATXT_INVALIDLAT                      5 + MATXTERR_OFFSET
#define     MATXT_INVALIDLON                      6 + MATXTERR_OFFSET
#define     MATXT_INVALIDCATEGORY                 7 + MATXTERR_OFFSET
#define     MATXT_INVALIDDATASIZE                 8 + MATXTERR_OFFSET
#define     MATXT_INVALIDSECURITYCLASS            9 + MATXTERR_OFFSET
#define     MATXT_INVALIDORIGINATINGSITE         10 + MATXTERR_OFFSET
#define     MATXT_INVALIDRECEIPTMETHOD           11 + MATXTERR_OFFSET
#define     MATXT_INVALIDOBJECTID                12 + MATXTERR_OFFSET
#define     MATXT_INVALIDSERVERNAME              13 + MATXTERR_OFFSET
#define     MATXT_INVALIDCONNECTIONNAME          14 + MATXTERR_OFFSET
#define     MATXT_INVALIDROLENAME                15 + MATXTERR_OFFSET
#define     MATXT_OBJECTNOTFOUND                 16 + MATXTERR_OFFSET
```

4 CONNECT APIS

The connect APIs are used to establish or disestablish a connection to the database server on which the MDTXT database resides.

Information about each API is presented in manual page format as follows:

NAME
Function Name – Provides a brief description of the function.

SYNOPSIS
Presents the calling syntax for the routine, including the declarations of the arguments and the return type. Also lists the necessary include files for each routine.

INPUT PARAMETERS
Describes each of the input parameters used by the function.

OUTPUT PARAMETERS
Describes each of the parameters output by the function.

DESCRIPTION
Describes what the function does and what events or side effects it causes.

RETURNS
Describes what the function returns.

NOTE
Provides any applicable notes about the function.

SEE ALSO
Provides a reference to related functions.

Examples showing the proper use of the APIs are presented in the *Textual Observation API Programming Manual*, referenced in Section 2.

4.1 MATXTConnect

NAME

`MATXTConnect` – Connects the application to the database.

SYNOPSIS

```
#include MATXTAPI.h

MATXTRET MATXTConnect( void );
```

INPUT PARAMETERS

None.

OUTPUT PARAMETERS

None.

DESCRIPTION

This subroutine establishes a default connection with the database server.

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTE

This interface (or `MATXTRemoteConnect`) must be called to establish a database connection before any other MATXT APIs may be called. It should only be called once per session.

SEE ALSO

`MATXTDisconnect`, `MATXTRemoteConnect`

4.2 MATXTDisconnect

NAME

`MATXTDisconnect()` – Disconnects the application from the database.

SYNOPSIS

```
#include MATXTAPI.h

MATXTRET MATXTDisconnect( void );
```

INPUT PARAMETERS

None.

OUTPUT PARAMETERS

None.

DESCRIPTION

This subroutine disconnects an application that connected using `MATXTConnect` from the database server, ending the database session.

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTE

`MATXTDisconnect` should be called to end each database session started by `MATXTConnect`.

SEE ALSO

`MATXTConnect`

4.3 MATXTRemoteConnect

NAME

`MATXTRemoteConnect` – Allows the application to connect to a textual observation database residing on a remote server and/or to establish multiple connections.

SYNOPSIS

```
#include      MATXTAPI.h

MATXTRET      MATXTRemoteConnect      ( char *pszServerName,
                                        char *pszConnectionName )
```

INPUT PARAMETERS

`char *pszServername` - Database server name (COE default is `$INFORMIXSERVER`)

`char *pszConnectionName` - Connection name for MDTXT database (`MATXT_DEFAULT_CONNECTION` is default)

OUTPUT PARAMETERS

None.

DESCRIPTION

The `MATXTRemoteConnection` routine allows an application to connect to a remote textual observation database server and/or to maintain multiple open connections on the same or different servers.

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTES

1. A database connection must be established using this interface or `MATXTConnect` (Section 4.1) before any other MATXT APIs may be called.
2. Passing NULL for either argument assumes the default setting for that argument.
3. Applications connecting to only one server and one database at a time should use the simpler `MATXTConnect` API.

SEE ALSO

`MATXTRemoteDisconnect`, `MATXTSetConnection`, `MATXTConnect`

4.4 MATXTRemoteDisconnect

NAME

`MATXTRemoteDisconnect` - Disconnects the application from a database residing on a remote server, or from one of multiple simultaneous connections.

SYNOPSIS

```
#include    MATXTAPI.h

MATXTRET MATXTRemoteDisconnect ( char *pszConnectionName );
```

INPUT PARAMETERS

`*pszConnectionName` – Name of connection to be disconnected.

OUTPUT PARAMETERS

None.

DESCRIPTION

This subroutine disconnects the application from the database connection specified, ending the database session.

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTE

`MATXTRemoteDisconnect` should be called to end each database session started by `MATXTRemoteConnect`.

SEE ALSO

`MATXTRemoteConnect`, `MATXTSetConnection`

4.5 MATXTSetConnection

NAME

`MATXTSetConnection` - Set the connection context for multiple connections to different textual observation servers and/or databases.

SYNOPSIS

```
#include MATXTAPI.h

MATXTRET MATXTSetConnection ( char *pszConnectionName );
```

INPUT PARAMETERS

`*pszConnectionName` – Connection desired for database transactions.

OUTPUT PARAMETERS

None.

DESCRIPTION

The `MATXTSetConnection` routine is used to set the connection context for multiple connections to different textual observation database servers and/or databases, making the specified connection the current connection.

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTE

Passing in NULL for the connection name assumes that the default connection (`MATXT_DEFAULT_CONNECTION`) is to be used.

SEE ALSO

`MATXTRemoteConnect`, `MATXTRemoteDisconnect`

5 RETRIEVAL APIS

This section describes the APIs that are used to retrieve data from the MDTXT database.

5.1 MATXTCatalog

NAME

`MATXTCatalog` – Retrieve catalog listing of textual observations meeting specified criteria.

SYNOPSIS

```
#include    MATXTAPI.h

MATXTRET    MATXTCatalog
(  MATXTCATQUERY     *pMATXTCatQuery,
                        int                 *pNumFound
                        MATXTLINKEDLIST     *pMATXTLLCatData
```

INPUT PARAMETERS

`MATXTQUERY *pMATXTCatQuery` - A pointer to a MATXTCATQUERY structure (Section 3.4.4) containing query criteria for the catalog retrieval.

OUTPUT PARAMETERS

`int *pNumFound` - The number of records found that matched the query criteria.

`MATXTLINKEDLIST *pMATXTLLCatData` - A linked list of `MATXTCATDATA` structures containing the catalog data retrieved.

DESCRIPTION

`MATXTCatalog` retrieves a catalog listing of textual observations in the database that match the query criteria provided in the input `MATXTCATQUERY` structure. The routine returns the number of matches found and a linked list of `MATXTCATDATA` structures (Section 3.4.5).

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTES

1. The application must establish a database connection using `MATXTConnect` (Section 4.1) or `MATXTRemoteConnect` (Section 4.3) before `MATXTCatalog` may be called.
2. `MATXTFreeLL` should be called to free the linked list upon completion of processing.

SEE ALSO

`MATXTConnect`, `MATXTRemoteConnect`, `MATXTFreeLL`

5.2 MATXTGetByQuery

NAME

`MATXTGetByQuery` - Retrieve a set of textual observations/bulletins from the database that match specified query criteria.

SYNOPSIS

```
#include MATXTAPI.h

MATXTRET MATXTGetByQuery        ( MATXTCATQUERY         *pMATXTCatQuery,
                                  int                   *pNumFound
                                  MATXTLINKEDLIST       *pMATXTLLObs );
```

INPUT PARAMETERS

`MATXTCATQUERY *pMATXTCatQuery` - Pointer to a `MATXTCATQUERY` structure containing the query criteria.

OUTPUT PARAMETERS

`int *pnNumFound` – The number of observations in the database that matched the query criteria.

`MATXTLINKEDLIST *pMATXTLLObs` – A linked list of `MATXTOBS` structures containing the retrieved text observation data.

DESCRIPTION

The `MATXTGetByQuery` routine takes as input a pointer to a `MATXTCATQUERY` structure containing the query criteria. It returns the number of matching observations found, a linked list of `MATXTOBS` structures containing the text observation data, and the `MATXTRET` return status structure.

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTES

1. `MATXTConnect` or `MATXTRemoteConnect` must have been called to start a database session before `MATXTGetByQuery` may be called. `MATXTFreeLL` should be called to free the linked list upon completion of processing.

SEE ALSO

`MATXTGetByID`, `MATXTFreeLL`, `MATXTConnect`, `MATXTRemoteConnect`

5.3 MATXTGetByID

NAME

`MATXTGetByID` - Retrieve a single textual observation or bulletin from the database given the `ObjectID`.

SYNOPSIS

```
#include MATXTAPI.h

MATXTRET MATXTGetByID ( int nObjectID, MATXTOBS *pMATXTObs );
```

INPUT PARAMETERS

`int nObjectID` - Object identifier for the textual observation to be retrieved, found using the `MATXTCatalog` method (Section 5.1).

OUTPUT PARAMETERS

`MATXTOBS *pMATXTObs` - A `MATXTOBS` structure containing the retrieved observation/bulletin data.

DESCRIPTION

The `MATXTGetByID` routine returns data from a single textual observation/bulletin, given as input the `ObjectID` of the record to be retrieved. The `ObjectID` may be found using the `MATXTCatalog` method. `MATXTGetByID` returns the observation/bulletin data in a `MATXTOBS` structure, and also returns the `MATXTRET` status structure.

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTES

1. The application must establish a database connection using `MATXTConnect` (Section 4.1) or `MATXTRemoteConnect` (Section 4.3) before `MATXTGetByID` may be called.

SEE ALSO

`MATXTConnect`, `MATXTRemoteConnect`, `MATXTCatalog`

(This page intentionally left blank.)

6 DATA MANAGEMENT APIS

6.1 MATXTIngest

NAME

`MATXTIngest` – Ingests a textual observation record into the database.

SYNOPSIS

```
#include MATXTAPI.h

MATXTRET MATXTIngest ( MATXTOBS *pMATXTObs, int *pnObjectID );
```

INPUT PARAMETERS

`MATXTOBs *pMATXTObs` - A pointer to a `MATXTOBS` structure (Section 3.4.3) containing the textual observation data to be ingested.

OUTPUT PARAMETERS

`int *pnObjectID` - Unique object identifier assigned to the textual observation record in the database.

DESCRIPTION

The `MATXTIngest` function ingests a textual observation into the database. It takes as input a pointer to a `MATXTOBS` structure containing the textual observation data, and returns a unique object identifier assigned to the observation in the database.

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTES

1. The application must establish a database connection using `MATXTConnect` (Section 4.1) or `MATXTRemoteConnect` (Section 4.3) before `MATXTIngest` may be called.

SEE ALSO

`MATXTConnect`, `MATXTRemoteConnect`

6.2 MATXTUpdateByID

NAME

`MATXTUpdateByID` – Update a textual observation/bulletin record in the database.

SYNOPSIS

```
#include MATXTAPI.h

MATXTRET MATXTUpdateByID ( MATXTOBS *pMATXTObs, int *pnObjectID );
```

INPUT PARAMETERS

`MATXTOBS *pMATXTObs` - Pointer to a `MATXTOBS` structure (Section 3.4.3) containing the data with which to update the record in the database

`int *pnObjectID` – Object identifier of the record in the database that is to be updated, found using the `MATXTCatalog` method.

OUTPUT PARAMETERS

None.

DESCRIPTION

`MATXTUpdateByID` updates a textual observation/bulletin record in the database with new data contained in the input `MATXTOBS` structure. The record to be updated is identified by the `ObjectID` specified. The routine returns a `MATXTRET` return status structure. This subroutine will not overwrite an object that has an `nDataCategory` field of `MATXT_BASE` (0). If this is the case, a new object will be ingested into the database with an `nDataCategory` of `MATXT_EDITED` (1), and the new `ObjectID` will be returned to the caller.

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTES

1. The calling application must have been connected to the database through a call to `MATXTConnect` or `MATXTRemoteConnect` before this routine may be called.

SEE ALSO

`MATXTIngest`, `MATXTDeleteByID`, `MATXTCatalog`, `MATXTConnect`, `MATXTRemoteConnect`

6.3 MATXTDeleteByID

NAME

`MATXTDeleteByID` – Delete a textual observation/bulletin record from the database.

SYNOPSIS

```
#include MATXTAPI.h

MATXTRET MATXTDeleteByID ( int nObjectID );
```

INPUT PARAMETERS

`int nObjectID` - Object Identifier for the record in the database that is to be deleted, found using the `MATXTCatalog` method.

OUTPUT PARAMETERS

None.

DESCRIPTION

The `MATXTDeleteByID` routine deletes a textual observation/bulletin record from the database, given the `ObjectID` of the record. It returns a `MATXTRET` structure indicating the status of the operation.

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTES

1. The calling application must have been connected to the database through a call to `MATXTConnect` or `MATXTRemoteConnect` before `MATXTDeleteByID` may be called.

SEE ALSO

`MATXTIngest`, `MATXTUpdateByID`, `MATXTCatalog`, `MATXTConnect`, `MATXTRemoteConnect`

6.4 MATXTDeleteByQuery

NAME

`MATXTDeleteByQuery` – Delete textual observations meeting specified criteria.

SYNOPSIS

```
#include MATXTAPI.h
```

INPUT PARAMETERS

`MATXTCATQUERY *pmatxtCatQuery` - A pointer to the `MATXTCATQUERY` structure (Section 3.4.4) containing the criteria used for the delete.

OUTPUT PARAMETERS

`int *pnNumDeleted` – The number of records deleted.

DESCRIPTION

`MATXTDeleteByQuery` deletes textual observations from the database that match the query criteria provided in the input `MATXTCATQUERY` structure. The routine returns the number of observations that were deleted.

RETURNS

`MATXTRET` Structure – See Section 3.4.6 for details.

NOTES

1. This API was developed to assist the purge program. It was introduced in version 4.2.0.0 of MATXT.

SEE ALSO

`MATXTConnect`, `MATXTDisconnect`

7 MATXT UTILITY METHODS AND FUNCTIONS

7.1 MATXTFreeLL

NAME

`MATXTFreeLL` – Free a textual observation linked list.

SYNOPSIS

```
#include MATXTAPI.h

MATXTRET MATXTFreeLL ( MATXTLINKEDLIST *pMATXTLL );
```

INPUT PARAMETERS

`MATXTLINKEDLIST *pMATXTLL` – Pointer to the linked list to be freed.

OUTPUT PARAMETERS

None.

DESCRIPTION

`MATXTFreeLL` frees a linked list that has been returned by calls to `MATXTGetByQuery` or `MATXTCatalog`. It returns a `MATXTRET` return status structure.

RETURNS

`MATXTRET` structure – See Section 3.4.6 for details.

NOTE

1. The application must free the head of the linked list after the call to `MATXTFreeLL`.

SEE ALSO

`MATXTGetByQuery`, `MATXTCatalog`

(This page intentionally left blank.)

8 NOTES

8.1 Glossary of Acronyms

AESS	Allied Environmental Support System
AOI	Area Of Interest
API	Application Program Interface
APIRM	API Reference Manual
COE	Common Operating Environment
DII	Defense Information Infrastructure
GCCS	Global Command and Control System
IC4ISR	Integrated Command, Control, Communications, Computer, and Intelligence Surveillance Reconnaissance
IMOSS	Interim Mobil Oceanographic Support System
IP	Installation Procedure
JMCIS	Joint Maritime Command Information System
JMS	Joint METOC Segment
LLT	Latitude-Longitude-Time
MATXT	Textual Observation API Segment of the TESS(NC) METOC Database

MDTXT	Textual Observation Database Segment of the TESS(NC) METOC Database
METOC	Meteorology and Oceanography
MIDDS	Meteorological Integrated Data Display System
NITES	Navy Integrated Tactical Environmental Subsystem
PM	Programming Manual
PS	Performance Specification
SQL	Structured Query Language
SVD	Software Version Description
TEDS	Tactical Environmental Data System
TESS(NC)	Tactical Environmental Support System Next Century